Application Note 4

Microprocessor Board Troubleshooting Techniques Using Processor Emulation (Pentium And Power PC)

Important: *It is assumed that the reader has a good knowledge of microprocessor board architecture, circuitry, and programming.*

1. Introduction:

Conventional troubleshooting techniques involves a combination of stimulating the defective circuit using a signal generator, and probing using an oscilloscope. Figure 1 shows a typical set up. The component at which the correct stimulus disappears is the defective component.

With microprocessor-based boards this technique is not possible, as there is no obvious place to connect a signal generator. In addition with large scale integration and high density packaging (e.g. BGAs), probing is often not possible. Alternatives methods are needed.

Some effective techniques are:

- BIT (built-in test): Test software is built-in to the board's boot ROM. Test results are output via an available output port (e.g. serial). Although an effective technique BIT has a number of disadvantages: results are generally only pass/fail; the board kernel (cpu, memory, some i/o) must be operational before BIT can be used; it must be built-in at design time.
- Logic Analyser: It is a useful method for monitoring board activity, however it is difficult to connect, and to be used effectively a high degree of skill is needed. Essentially, an engineers tool!
- Emulator: An emulator, such as International Test Technologies MT2000, combines functional testing, stimulus generation, and signal monitoring. An emulator takes control of a microprocessor-based board, and allows the user to fully control the test process. Even completely 'dead' boards can be controlled! It is the most flexible method for microprocessor board troubleshooting, and little or no design-for-test is needed.

2. Overview of Troubleshooting Steps Using an Emulator:

As with all engineering tasks, breaking the troubleshooting process into manageable steps is the best approach. The recommended approach consists of three top-level steps:

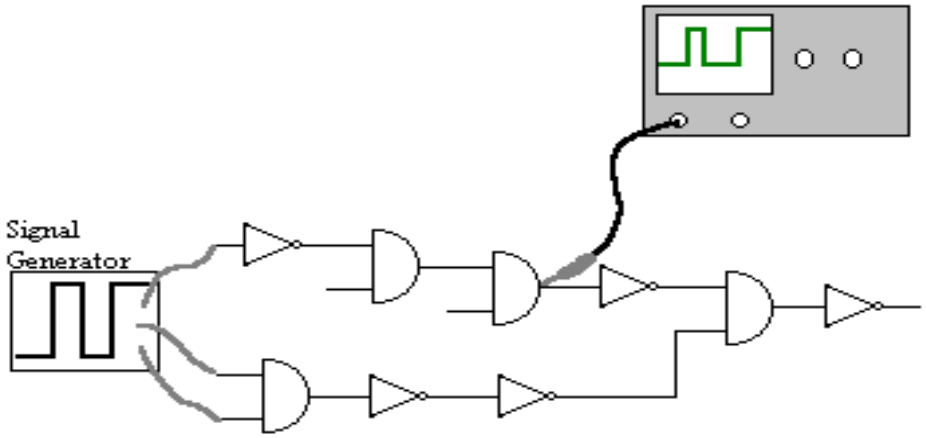

Fig. 1: Conventional Circuit Troubleshooting

1 Understand the board structure. That is, divide the board into functional blocks, and create a hierarchical block diagram of the board structure, similar to the one shown in Fig. 2 (as an example this shows the familiar PC architecture, but the same principles apply to any board architecture).
2 Starting at the processor, and working through the board hierarchy, devise tests to functionally verify each functional block. This will isolate the fault area to a particular functional block.
3 Once the fault area has been isolated the actual defect can be identified using a combination of visual inspection and probing.

3. Step 1 - Understanding Board Structure:

Using available board documentation divide the board into functional blocks, and draw a block diagram of the board structure, similar to the one shown in Fig. 2.

To do this:
1 Identify board processor.
2 Identify major buses and their relationships (processor bus, PCI, ISA, PMC, CompactPCI, etc.).
3 Identify any bus bridges (a bridge is an IC which converts one bus type to another bus type)
4 Identify board memory and assign addresses.

Identify major peripheral devices, I/O components which drive them, and the buses which connect to the

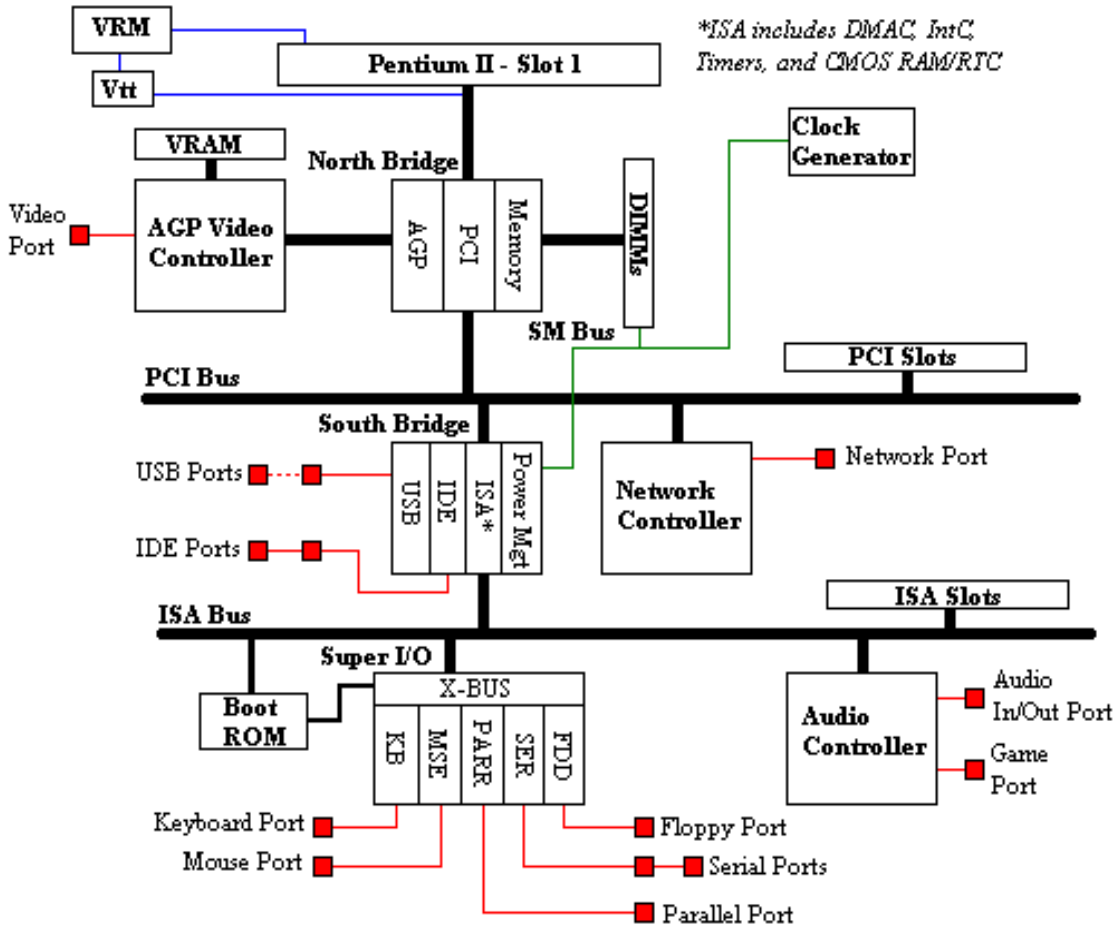

Fig. 2: Example Board Architecture

5 I/O devices (The best way to do this is to start at the I/O connector and work backwards). Assign I/O addresses to each I/O device.

Once, the block diagram has been created, convert this into a troubleshooting tree, as shown in Fig. 3. The tree shows an order in which to troubleshoot the board (from the root, and on down through the various levels), and explicitly shows how the failure of one functional block high up in the hierarchy, can fail blocks lower in the hierarchy. Therefore, a test failure is only considered valid, if all devices between the failed device and the root have passed, otherwise the failure can be ignored as the failed device higher in the tree has caused the current failure.

Lastly, if available, collect the datasheets for all the major board ICs. The best source for these are the websites of the IC manufacturers.

4. Step 2 - Fault Isolation:

Once the functional blocks have been identified, and a troubleshooting tree has been created, testing should commence from the processor down. To verify each functional block a test sequence is created using the emulator's pre-programmed functions. For this discussion, functional blocks and the test sequences used can be categorised as follows:

- Processor Area
- Buses
- Bus Bridges
- Memory
- Input/Output

Note: MT2xx0 Board Test Program Options are shown as {x menu, y option}. Alternatively, all these options have programming equivalents; these are discussed in Applications Note #5.

4.1 Processor Area:

If the processor can reset and enter test mode (i.e. 'the failure to initialise testing' message is not displayed), it is generally safe to assume that the processor area is OK.

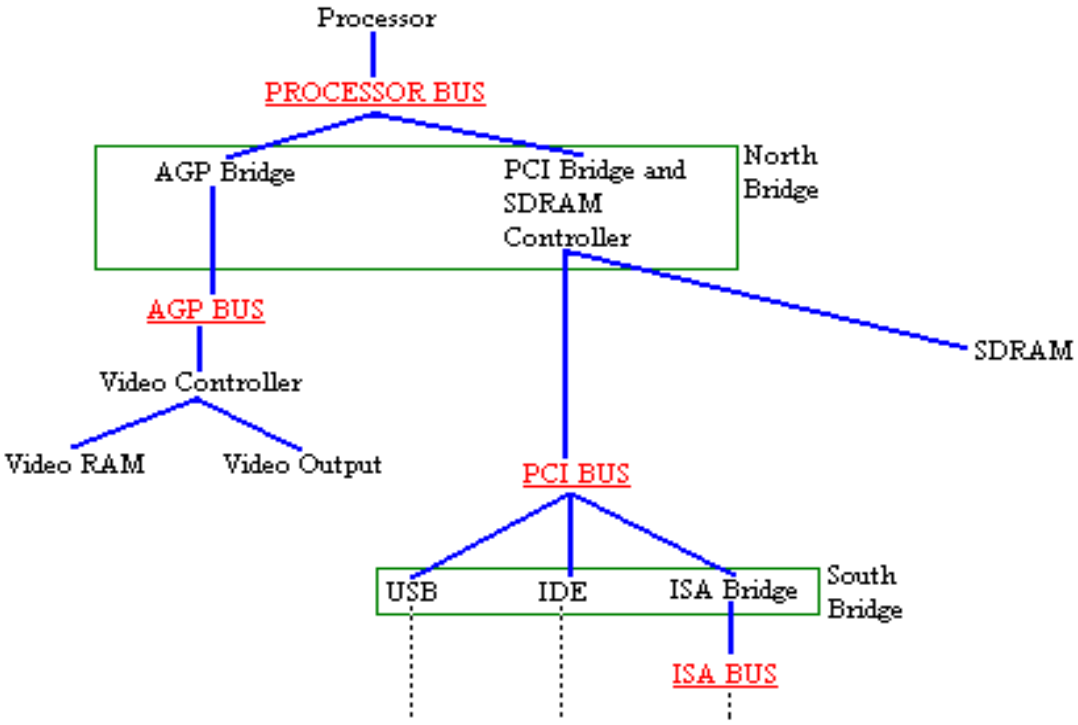

Fig. 3: Section of Troubleshooting Tree

If the processor fails to enter test mode, you should run the MT2000 CPU signals test *{test menu, CPU signals test option}*. This will verify clock, reset, and any signals which might stop the processor operating. Any defective signals identified can then be traced using an oscilloscope. If the pins test passes the following steps are recommended:

1 Verify power lines using oscilloscope or DMM.
2 Verify clock using oscilloscope, as the clock test using the pins test, is only a presence detect, not a full check.
3 Using stimulus mode *{stimulus menu, stimulus option}*, generate a repeated reset, and check processor BRDY signal. Lack of activity or a stuck-low indicates a problem.
4 If all of the above pass, try replacing the CPU and/or main chipset (e.g. northbridge).

4.2 Buses:

To diagnose bus defects (stucks, opens, shorts, etc.) the MT2xx0 provides a number of automated tests which are accessible from the *{test menu, bus tests option}*.

Note: For all bus tests you must set the 'enter debug mode with reset' option. Select from *{edit menu, reset option}*.

The automated bus tests are:

ROM Bus Test

Select this from *{test menu, bus tests option}*. This test verifies the operation of the buses from the processor to the boot ROM, and diagnoses most data and address bus problems (except shorts). On selection, enter the address range of the boot ROM, and select test. Any defective data or address bus lines found are reported by bus type and line.

PCI Bus Test (pentium only)

Select this from *{test menu, bus tests option}*. A PCI bus test card must be connected to a free PCI slot before this test can be executed. On execution a series of binary test patterns is sent across the PCI bus. These are captured by the test card, and sent to the Host PC for analysis. Both bus verification and diagnosis is performed, and any defective data, address, or control lines are reported. Both processor bus and PCI problems are detected and diagnosed using this option.

ISA Bus Test (pentium only)

Select this from *{test menu, bus tests option}*. An ISA bus test card must be connected to a free ISA slot before this test can be executed. On execution a series of binary test patterns is sent across the ISA bus. These are captured by the test card, and sent to the Host PC for analysis. Both bus verification and diagnosis is performed, and any defective data, address, or control lines are reported. Only ISA bus problems are detected using this option.

4.3 Bus Bridges:

A bus bridge converts one bus type to another, e.g. processor bus to PCI bus. To test and diagnose a bridge use the following methods:

1 All bridge chips contain programmable read/write registers. A read/write test to these registers will verify whether the chip is accessible to the processor. (see Applications Note #5 for more detail on read/write testing)
2 The ability to access devices (read or write to memory or i/o) on the other side of the bridge verifies that the bridge is performing its intended function.

4.4 Memory:

The MT2xx0 comes supplied with a number of pre-programmed RAM tests. To select, use *{test menu, RAM tests option}*.

The available RAM tests are:
RAM Bus Test
Verifies the buses up to the RAM chips. Bad data and address lines are automatically diagnosed and reported.
Short RAM Test (pentium only)
Verifies the buses up to the RAM chips, and that all RAM cells are readable and writeable. Bad data and address lines, and RAM cells are automatically diagnosed and reported.
Long RAM Test (pentium only)
Performs a more pattern sensitive RAM test. Use this, if the other RAM tests pass and you still suspect the RAM as being the problem.
DRAM Refresh Test (pentium only)
Verifies that all DRAM cells are refreshing correctly.

To test RAM just enter the address range and click on test.

Often RAM is controlled by a memory controller. This devices contains internal registers which must be initialised before the RAM is visible to the processor. There are two ways to do this.

1. Using the boot ROM BIOS. Select the option to run the MT2xx0 without a reset *{edit menu, reset option}*, then reset the board and let the BIOS initialise the memory controller. Any of the RAM tests can then be used. **BEWARE!** On the defective boards the BIOS may not run correctly so the memory controller may not be initialised correctly, and the memory will fail the RAM tests. However, the RAM may not be the problem, as the BIOS may not have run far enough to initialise the memory controller! Only use this method, if you are confident that the BIOS has made it to the point where it has initialised the memory controller.
2. Initialise manually. Use the MT2xx0 Read/Write I/O option *{test menu, read/write i/o}* to manually initialise the memory controller (see Application Notes #5 for details on how to do this).

4.5 Input/Output:

Input/Output ICs can be sub-divided into two parts. The section that interfaces to the processor and its buses, and the part that interface to the outside world. Different test methods are used for each. More details on I/O testing are presented in Applications Note #5.

4.5.1 Input/Output IC Access:

Most input/output chips contain internal registers which can be accessed by the processor's read/write instructions. Using the *{test menu, read/write i/o option}*, test patterns can be written and verified to these registers. To do this select a read/writeable register and enter its ports address and write and verify a series patterns such as AA and 55, to the port. Alternatively, if the chip has an ID register, just verify the ID using a read I/O command. Such registers will verify that the processor and its buses can access the i/o chip. More detailed diagnostic information can be derived using the *{test menu, i/o bus test option}*.

4.5.2 Outside World (Peripheral) Access:

Note: This is a complex topic, requiring a test program to be developed, therefore it is covered in more detail in Applications Note #5.

This test must verify that the I/O device can communicate with the external device under its control. It consists of the following steps:

1 Initialise the I/O device to enable it to communicate with the external device. This involves writing specific values to some of the Input/Output IC's internal registers.
2 Send and verify data to the external device.
3 Receive and verify data from the external devices

Data is sent and received to/from the external device using some of the Input/Output IC's internal registers. Data to/from the external device is verified by using one of the following methods:

1 A real peripheral such as a disk drive is connected, and data is written, then read back and verified.
2 A loopback is used. This wraps back transmitted data into the receive channel, allowing both channels to be simultaneously verified.
3 An external instrument is used to generate or measure test signals (e.g. an oscilloscope is used to verify a generated audio signal).

5. Step 3 - Fault Identification:

The defective functional blocks isolated using the methods described above, will probably still contain a few components and numerous interconnects. To identify the actual defect within the failed functional block use the following methods.

5.1 Visual Inspection:

Keep it simple! Before returning to the emulator, perform a detailed visual inspection of the isolated area. Look for shorts, bad solder joints, 'tombstoned' resistors, wrong ICs, etc.

5.2 Tactile Inspection:

Again, keep it simple! Are there any overheating ICs?

Now return to the emulator. Loop the failing test *{run menu, loop mode}*. Press down on all major ICs. Does the test now pass. This will detect open circuits on devices such a BGAs etc.

5.3 Probing:

Use loop mode, or *{stimulus menu, all options}* to inject known good signals into the defective area. Probe (where possible) using an oscilloscope or the MT2xx0 logic probe. Start by just look for wrong levels, lack of activity, or tri-state conditions. If necessary, create the same stimulus on a good board and compare.

6. Summary:

An emulator provides the best technique for troubleshooting microprocessor-based boards. By following a simple 3-step procedure, defects can be rapidly isolated.

1 Understand board structure. Break it into functional blocks and create a troubleshooting tree.
2 Isolate the fault area by using simple test sequences to verify each functional block.
3 Finally identify the actual defect using a combination of inspection and probing.

Happy Troubleshooting!

Further Information

For further information or assistance on this application note please contact:

International Test Technologies
First Floor,
Winton House,
Stoke Road,
Stoke-On-Trent,
Staffs, ST4 2RW
England

Tel: +44 1782 418020
Fax: +44 1782 418025

e-mail: support@intertesttech.com
Web site: www.intertesttech.com